AMENDMENTS TO THE SPECIFICATION:

Please delete the paragraph at page 3, lines 9-21 from the application as follows:

~~It is also known to detect data anomalies in the form of vulnerabilities in compiled binary code and disassembled binary code using hand-crafted rules to identify potential bugs in the code. The rules are generated by human experts in vulnerability detection. For example, in a hand crafted rule-set category, a "*SmartRisk Analyzer*" product of the @stake company looks for "triggers" in a computer program written in assembly language code. "Triggers" are calls to functions (such as *strcpy*) known to be vulnerable. On finding a trigger, *SmartRisk Analyzer* traces a data and control path back through the program in order to determine possible values of parameters comprising an argument of the vulnerable or unsafe function, to see if the function call will be vulnerable during run time. So-called black-box testing technologies are more commonly used, , usually referred to as "fuzzers"; fuzzers essentially perform a random search or a brute force search through a (usually intractably large) space of test vectors. They can also be enhanced by hand crafting constraints on the search space's domain.~~

Please amend the paragraph beginning at page 3, line 22 as follows:

As before, the need for human experts to generate rules is undesirable because it is onerous. Although human experts may have much experience, it is not feasible for them to learn from all possible scenarios. Gaining additional and wider experience takes time and resources. Once a rule base is derived, it can be used to identify whether new software applications contain potentially exploitable binary code. However, current systems of vulnerability detection have rule bases which are typically static, i.e. unchanging over time unless rules are added or edited manually. As new vulnerabilities become apparent, such a system needs to be updated by hand in order to be able to identify associated 'bugs'. Further deficiencies of a rule-based approach~~, such as that used by @Stake,~~ is that it has a limitation on 'semantic depth' that is practical for such techniques. A vulnerability having semantics which are sufficiently complex is not likely to be detected by such an approach.

Please insert the following new paragraph after the paragraph ending on page 34, line 18:

Data anomalies may be detected in the form of vulnerabilities in compiled binary code and disassembled binary code using hand-crafted rules to identify potential bugs in the code. The rules are generated by human experts in vulnerability detection. For example, in a hand crafted rule set category, a "*SmartRisk Analyzer"* product of the @stake company looks for "triggers" in a computer program written in assembly language code. "Triggers" are calls to functions (such as *strcpy*) known to be vulnerable. On finding a trigger, *SmartRisk Analyzer* traces a data and control path back through the program in order to determine possible values of parameters comprising an argument of the vulnerable or unsafe function, to see if the function call will be vulnerable during run time. So-called black-box testing technologies are more commonly used, , usually referred to as "fuzzers"; fuzzers essentially perform a random search or a brute force search through a (usually intractably large) space of test vectors. They can also be enhanced by hand crafting constraints on the search space's domain.